A Regret Minimization Approach to Frameless Irregular Repetition Slotted Aloha: IRSA-RM

Iman Hmedoush, Cédric Adjih, Paul Mühlethaler

A Regret Minimization Approach to Frameless Irregular Repetition Slotted Aloha: IRSA-RM

Iman Hmedoush, Cédric Adjih, and Paul Mühlethaler

Inria, France
<first name>.<last name>@inria.fr

Abstract. Wireless communications play an important part in the systems of the Internet of Things (IoT). Recently, there has been a trend towards long-range communications systems for the IoT, including cellular networks. For many use cases, such as massive machine-type communications (mMTC), performance can be gained by moving away from the classical model of connection establishment and adopting random access methods. Associated with physical layer techniques such as Successive Interference Cancellation (SIC), or Non-Orthogonal Multiple Access (NOMA), the performance of random access can be dramatically improved, giving rise to novel random access protocol designs. This article studies one of these modern random access protocols: Irregular Repetition Slotted Aloha (IRSA). Since optimizing its parameters is not an easily solved problem, in this article we use a reinforcement learning approach for that purpose. We adopt one specific variant of reinforcement learning, Regret Minimization, to learn the protocol parameters. We explain why it is selected, how to apply it to our problem with centralized learning, and finally, we provide both simulation results and insights into the learning process. The results obtained show the excellent performance of IRSA when it is optimized with Regret Minimization.

Keywords: IRSA · Regret Minimization · Random Access

1 Introduction

1.1 Communications in the Internet of Things

In recent years there has been an increase in the technological demands on embedded systems and sensors, that led to the emergence of the Internet of Things (IoT). The Internet of Things provides a comprehensive set of solutions that enables the seamless interconnection of a smart community of devices and sensors. This article focuses on one of the most important challenges for IoT networks: communications. This has been a highly visited topic for research, development, and standardization over the past decade because the differences in IoT applications have resulted in a large variety of requirements and constraints. These constraints include the ability to support millions of connected devices, the necessity for low power consumption, and the need for high throughput, low latency, and high reliability.

In the IoT, there has been a recent trend towards using long-range low power networks: with cellular networks (in 4G with NB-IoT, and LTE-M, in 5G with URLLC, and NR-light in future 5G Rel. 17), or with networks in the unlicensed band (LoRaWAN, SigFox). As a consequence, there have been new directions in the design of IoT protocols to satisfy the critical requirements of IoT applications, including modern variants of random access methods. One such method is the focus of this article.

1.2 Modern Random Access Protocols for IoT Communications

A recent family of random access protocols has emerged as a promising solution for modern random access: Irregular Repetition Slotted Aloha (IRSA)[4], and its generalization with coding, Coded Slotted Aloha (CSA)[5]. It has become the focus of the IoT protocol designers' attention, since it has been shown that it could asymptotically reach the optimal throughput of one retrieved packet per slot, in the classical random access collision model (where the maximum throughput of slotted ALOHA is $\frac{1}{e}$). NOMA variants like PDMA/IRSA [6] exist.

The principle is that the users send multiple copies of each of their data packets to the receiver, which uses Successive Interference Cancellation (SIC) to resolve the collisions. The transmission is done in slots. Each copy (also known as a replica) contains the same payload and the same preamble with additional information about the positions of its copies in other slots. Once one packet has been received without collision, SIC exploits this information to reconstruct the physical signal that corresponds to the decoded packet and subtracts this physical signal at the positions of its other copies.

1.3 Irregular Repetition Slotted Aloha (IRSA)

IRSA is a recent member of the CSA family that is an optimization of Contention Resolution Diversity Slotted Aloha (CRDSA)[11]. In CRDSA, the users would repeat their packets twice. In IRSA, the users are allowed to choose the number of repetitions (*repetition degree*) according to a probability distribution. In classical IRSA, we consider a MAC frame of M slots and N users who send their packets towards a central node. The channel load is defined as $g = \frac{N}{M}$ (i.e. average number of users per slot). The receiver uses SIC to decode the collided packets. Each user chooses its repetition degree based on the user degree distribution $\Lambda = (\Lambda_0, \Lambda_1, \dots \Lambda_D)$, where Λ_i is the probability of using the degree i and D is the maximum degree.

At the end of each frame, the receiver starts performing iterative decoding using SIC. At each decoding iteration, the receiver starts by searching for the non-colliding packets (referred to as the singletons). After finding, decoding, and recovering all the singletons in the frame, the receiver removes their physical copies from their positions in the frame. This suppresses some collisions and in turn, can make new singletons appear in the next decoding iteration. The iterative decoding continues until the receiver can not find any new singletons or the whole frame is decoded. Fig. 1 represents a simple example used to illustrate

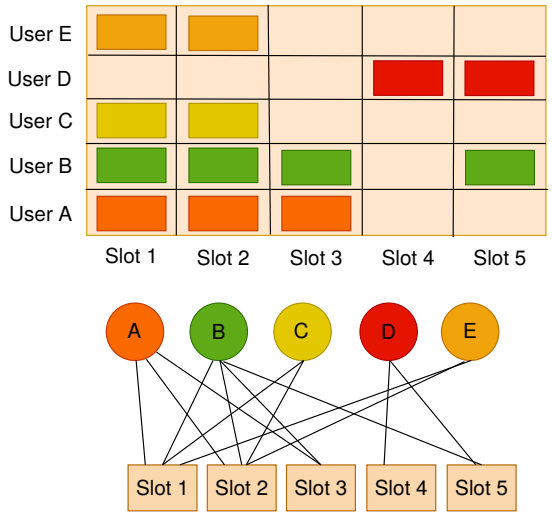


Fig. 1: IRSA representation: transmissions of users in slots (top), coding theory representation to model the decoding process (Tanner graph, bottom). Notice that transmissions are in the same frequency channel, hence when two users are transmitting on the same slot, there is a collision.

the SIC decoding. The figure shows a frame where 5 users compete to send their packets on 5 slots. The receiver starts by searching for the singletons. As can be seen from the figure, slot 4 has only one packet from user D, so it can be decoded on slot 4. Then, with SIC, it can be subtracted from slot 5. This allows the packet from user B to be a new singleton on slot 5 so that it is decoded from slot 5 and subtracted with SIC in slots 1, 2, and 3. Now the packet of user A becomes a new singleton on slot 3 so it is decoded on slot 3 and subtracted from slots 1 and 2 as well. We can see that the packets of users C and E cannot be retrieved since none of their replicas had ever become a singleton and they form a stopping set. At this point, the decoding process stops.

The remainder of this paper is organized as follows: section 2 introduces related work on applying machine learning algorithms to random access protocols. Then, section 3 explains the system model and our problem statement. In section 4, we introduce our learning approach: IRSA-RM, and detail the algorithm structure. Finally, we present our numerical results in section 5 and conclude in section 6.

2 Related Work

In the literature, several research directions have addressed topics that are related to modern random access protocols. Naturally, there also exists extensive literature on random access protocols themselves that date back over several decades. In this section, we focus on modern random access, and on research

studies that applied various reinforcement learning techniques. We identify the following related topics where machine learning techniques have been used: cognitive networks with spectrum sensing, classic random access protocols in IoT networks, and finally, more specific machine learning approaches to protocols of the IRSA family itself, or NOMA-based protocols. In the following, we describe some of the related articles that have covered these topics. In cognitive networks, Dynamic Spectrum Access is a wireless network paradigm where the users exploit their knowledge of the environment in order to successfully access a shared medium and maximize their throughput. The problem of dynamic spectrum access for wireless networks has been recently explored with machine learning techniques in [12] and [13]. It is shown that the problems of joint user association and spectrum access are typically combinatorial and non-convex, and require near-complete and accurate information to obtain the optimal strategy. According to [14], developing efficient learning approaches to optimize medium access has been the center of attention of many research works. In particular, Deep Q-Learning (DQL) provides promising solutions for the Dynamic Spectrum Access problem specifically, for IoT networks. In [15], a novel distributed dynamic spectrum access algorithm based on deep multi-user reinforcement learning (DRL) has been proposed. The users transmit over shared channels using a random access protocol. Time is slotted but no SIC is used in the receiver to resolve collisions. In the proposed approach, every user maps their current state (the history of selected actions and past network state observations) to a certain action (shared orthogonal channel) based on a trained deep-Q network. Their objective is to maximize a utility function. The proposed algorithm enables the user to learn good policies in an online distributed manner. The authors of [16] address the problem of collisions and idle time of random access protocols by designing a fully distributed IoT protocol to improve the device access to the shared medium. The proposed online learning scheme is based on designing optimized dictionaries of transmission patterns to avoid collisions between users. The dictionary contains a subset of the possible binary vectors of a length equal to the total number of slots in the frame. The goal is to select an optimized set of transmission patterns from the dictionary, where the dictionary is common to all users. The scheme provides some URLLC guarantees for IoT applications that require the same time latency, energy efficiency, and low coordination overhead. Dynamic multi-channel access was also considered in [9], where the user selects a channel, at each time slot, from multiple correlated channels. Each user can observe the state of the chosen channel only at a given time slot, which means that the current state of the system is not fully observable, hence the problem is modeled as a Partially Observable Markov Decision Process (POMDP). The aim of the study is to design an adaptive DQN framework that can adapt to time variations and maximize the long-term expected reward for each user.

Another work direction for designing efficient IoT protocols is to enhance existing MAC protocols so that they fit the new requirements of IoT networks. Optimizing the performance of MAC protocols has been addressed in many research studies over the last forty years. Some of these studies have introduced

machine learning techniques to variants of the ALOHA protocol family. A novel Q-learning based on Informed Receiving Protocol has been introduced in [3]. ALOHA-QIR provides some intelligence to the nodes to access the slots that have a lower probability of collision. The nodes keep hopping to different slots to learn the optimum ones. In this ALOHA variant, the nodes keep listening during the hopping while the receiver is informed by the preferred slots of each node by sending "ping packets", so the receiver can turn off when needed. The classical Q-learning algorithm with a simple reward design (± 1) is used to learn the optimum slots to select. The proposed approach helps to achieve over twice the maximum throughput of Slotted ALOHA. In [17], the IRSA MAC protocol is optimized using online learning. By considering the base station as the decision-maker, the performance of IRSA is optimized by maximizing a utility function that reflects the number of decoded packets. The problem of optimal resource allocation (slots allocation) is formalized as a Multi-Armed-Bandit (MAB) problem. The authors use the Bayesian UCB algorithm to solve the MAB problem and compare it with other commonly used methods. The degree distribution is also optimized by fixing the degrees and optimizing the probabilities of selecting them (e.g., of the form $\Lambda_2, \Lambda_3, \Lambda_8$).

3 System Model and Assumptions

3.1 System Description

We consider IRSA as an access protocol for users (devices) sharing a communication channel to a receiver. We assume that the receiver is a single base station. As for other protocols of the CSA family, the access time of the channel is divided into slots of equal duration. The duration of the slot is equal to the time needed to transmit a packet (including propagation delays, etc.). In our system, however, we assume that a *frameless IRSA* is used (as in [7]), as opposed to the framed, classical version of the IRSA protocols. In framed IRSA, there is a frame of a predefined length, where each user randomly selects slots, and at the end of which the decoding is performed. In frameless IRSA, there is a very large set of slots, potentially infinite. In our model, for practical reasons, this set of slots always has a fixed size of M slots and it is called a contention round. We divide the contention round into virtual frames where each virtual frame size is about 20% of the contention round. When the user decides to send a packet, the user is associated with a virtual frame. The active user sends the replicas of its packet only during the virtual frame period. The goal of introducing the virtual frame is to facilitate the decoding process and rewards computations, which will be explained more precisely in the next section. At each time slot, the number of active users is determined by a Poisson arrival rate μ (e.g. the number of active users on one slot is a random variable N_a with distribution $\Pr(N_a = k) = \frac{\mu^k e^{-\mu}}{\mu!}$). In our case, to be consistent with the literature, the arrival rate μ is also denoted network load G. It is the average number of active users on one slot. Each active user selects a repetition degree to use from a set

of multiple allowed degrees which are identical for all users. At the base station, SIC is used to resolve the collisions. Figure 2 shows the frameless IRSA structure

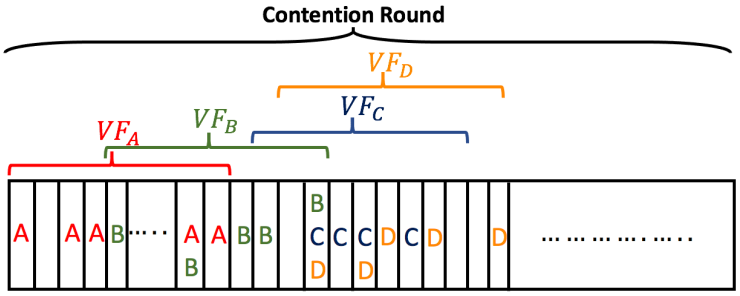


Fig. 2: IRSA frameless structure

with all active users and their associated virtual frames where they are allowed to send their packets. Virtual frames can overlap and the transmissions from different active users can cause, as seen in the figure. Unlike the classical IRSA decoding, (explained in section 1.3), the base station performs online decoding by decoding each received slot instead of waiting for the whole frame to end and then start the decoding process.

3.2 Problem Statement

Many studies have explored and analyzed the performance of IRSA in an IoT network. The main purpose of these studies is generally to find and study a better variant of this protocol. Given any IRSA system, the goal is often to find an optimized user degree distribution that maximizes a certain metric (throughput, achievable load, etc). This problem is usually formulated as an optimization problem which can be described as follows:

$$\begin{aligned} \underset{(\Lambda_i)}{\text{maximize}} \quad & C(\Lambda_0, \Lambda_1, .., \Lambda_D) \\ \text{subject to} \quad & 0 \leq \Lambda_i \leq 1 \ \forall i \\ & \sum_{i=0}^{i=D} \Lambda_i = 1 \end{aligned} \tag{1}$$

where C is the system criteria that needs to be optimized and D is the maximum degree. Depending on the system model, more constraints can be added to the optimization problem. Notice that the system is initially a stochastic optimization problem, as the performance depends on the random variables of the users' arrivals and their degree selections. Pioneering work in [4] adapted the Density Evolution (DE) method to analyze the asymptotic performance of different

(framed) variants of the IRSA protocol. It is based on an analogy with LDPC codes for which DE was initially introduced, as a tool for analyzing the asymptotic network capability to approach the error-correcting codes [18]. DE makes it possible to evaluate how many packets will be decoded, through iterations of functions, by modeling the decoding process steps: this is valid asymptotically when the frame size grows to infinity. Then one can compute the function C of the problem in Eq. (1) (through function iterations).

For several IRSA variants, the problem may still be difficult to solve, for example, in the case of the non-convexity or non-linearity of the constraints. But the optimization problem formulation may be useful. For instance, in our prior work [19], we revisited the variant K-IRSA proposed elsewhere: K-IRSA is a variant of classical IRSA with multiple packet reception capability at the receiver. Using a variant of the previous optimization problem Eq. (1), DE was used to write a new constraint on the edge degree distribution and the system criterion was to maximize the achievable system load. The OP was solved by converting the new constraint into a finite set of linear constraints and using the bisection method on linear problem formulations.

4 IRSA-RM: IRSA Based on Regret Minimization

4.1 Problem Formalization

We adopt the frameless IRSA structure (explained in section 3.1). For simplicity of presentation, we formulate the problem with two classes of users. The two classes have different access priorities. The users of the same class share the same degree distribution (Λ_i). The base station uses SIC to perform the slot by slot online decoding. Our objective is to find the best degree distribution for a known Poisson arrival rate $\mu = G$, that maximizes the weighted throughput of both classes. Formally, our problem could be written as an optimization problem using Eq. (1):

$$\begin{aligned} \underset{(\Lambda_i)}{\text{maximize}} \quad & \alpha_0 T_{C_0} + \alpha_1 T_{C_1} \\ \text{subject to} \quad & 0 \leq \Lambda_i \leq 1 \ \forall i \in [0, 1, 2, .., D] \\ & \sum_{i=0}^{i=D} \Lambda_i = 1 \end{aligned} \tag{2}$$

Where: T_C is the throughput of the class C and α_0, α_1 are constant weights indicating the importance of the throughput for each class.
One can think of solving such a problem by using the DE tool since frameless IRSA transmission and decoding using SIC can still be represented by a bipartite graph [7]. Since the BS performs the decoding for each slot, a part of the bipartite graph will be available at any time, thus the density evolution equations will not necessarily represent the decoding state in the middle of the contention round. Because classical DE [4] is valid only asymptotically and the finite length analysis can be computationally expensive, we adopt another direction in this article. We

propose a new learning framework for optimizing the transmission strategy of frameless IRSA. We consider a method of offline learning. We assume multi-agent settings and we apply the method of Regret Minimization, where each user wants to minimize its regret by taking better next decisions.

4.2 Reinforcement Learning Approaches and Regret Minimization

In this article, we use *Reinforcement Learning* (RL) to find good solutions of the problem Eq. (2). A classic reference on reinforcement learning in general is [8]. As in many network problems, the decisions taken by one node, device, or one user can be modeled as a Markov Decision Process (MDP) [8, section 3]: each participant of in the network is an *agent*, that makes decisions, denoted as *actions*, based on some current *state* from the environment. *Rewards* for each taken action are computed and are used to adjust the future choice of actions. Classical algorithms such as Q-Learning [8, section 6.5], Multi-Armed Bandits [8, section 2], and others, are well-known.

Applying those to random access introduces several challenges: the first one is that there are several agents instead of just one (*Multi-Agent* Reinforcement Learning, MARL, see [8, section 15.10]); the second one is that, by definition of random access, each agent only knows part of the network state, if only because it does not know the actions of other agents (*Partial Observable* Markov Decision Process, POMDP, see [8, section 17.3]).

Learning in a multi-agent setting is indeed a complex task: the impact of the decision taken by one agent may depend on the decisions taken by other agents in the system. Thus, first, classical RL approaches for a single agent can create difficulties, such as non-stationarity and oscillations when applied to multi-agent systems. Second, controlling multiple agents poses additional challenges compared to single-agent systems such as the definition of the collective goal of the agents, the heterogeneity of the agents, the ability to operate with a large number of agents, and partial observability [20].

Frameless IRSA is such a multi-agent system, subject to partial observability. Numerous learning approaches have been proposed in the literature to handle POMDP, including Deep Reinforcement learning (DRL) [20]. Many of the algorithms proposed in the literature lose their proof of convergence in a MARL setting, and there does not necessarily exist a general theory characterizing the cases under which every MARL algorithm is successful [21]. Their convergence or non-convergence dynamics is a topic of study by itself, with also strong links with game theory [21, 1]. Indeed, while applying Q-Learning to frameless IRSA, we experienced non-convergence, which led us to select an algorithm whose multi-agent dynamics have been well studied: Regret Minimization (RM) [2].

Regret Minimization is an algorithm where each agent maintains a set of *weights* for actions. Once normalized, the weights indicate the probability that the agent selects each action. At a given time, after the action selection by one agent according to weights, each such action i changes the environment state and has a corresponding reward which is provided by the environment. At the same given time, an optimal action could have been played by the agent instead,

which would have resulted in an optimal reward r. The difference between the optimal reward r_{opt} and the actual reward r_i gives the loss of the agent at that given time: $\ell_i = r_{\text{opt}} - r_i$, which is a measure of *regret* for selecting action i.

The Polynomial Weights algorithm is one of the Regret Minimization algorithms that assigns weights for each action and uses the "loss" concept to update the weights after each playing round [2]. Formally, it is as follows [2, page 13]:

$$\begin{aligned}
&\text{Initially: } w_i^{(1)} = 1 \text{ and } p_i^{(1)} = \frac{1}{|X|} \text{ for } i \in X \\
&\text{At time } t-1\text{: an action } i \in X \text{ is selected according to } (p_j^{(t-1)})_{j \in X} \\
&\qquad \text{the reward of action } i \text{ is computed: } r_i^{(t-1)} \\
&\qquad \text{the potential reward of the best action is: } r_{\text{opt}}^{(t-1)} \\
&\qquad \text{the loss is computed as: } \ell_i^{(t-1)} = r_{\text{opt}}^{(t-1)} - r_i^{(t-1)} \\
&\text{Update for time } t\text{: } w_i^{(t)} = w_i^{(t-1)}(1 - \eta \ell_i^{(t-1)}) \\
&\qquad p_i^{(t)} = \frac{w_i^{(t)}}{\sum_{i \epsilon X} w_i^{(t)}}
\end{aligned} \tag{3}$$

with:
X: the set of possible actions.
$w_i^{(1)}$: the initial associated weight to the action i.
$w_i^{(t)}$: the associated weight to the action i at time t.
$p_i^{(1)}$: the initial probability of using an action i out of $|X|$ actions.
$p_i^{(t)}$: the probability of using an action i at time t.
η: the learning parameter (akin to a learning rate).
The weights update is based on two main parameters: the learning parameter to control the speed of the weight changes, and the loss which specifies the impact of the played action by computing how far the played action was from optimality. The weights of the actions are used to compute the probability of using each of the actions in the next playing round.

Notice that richer variants of Regret Minimization have been proposed, such as Counterfactual Regret Minimization (CFR) [10]; with IRSA, they would be well suited for agents with richer interactions, for instance, agents taking decisions on the transmission of each replica (instead of selecting a degree once).

4.3 Applying of Regret Minimization to Frameless IRSA

Returning to our initial problem, we assume that the network consists of users competing in the same slotted wireless channel to transmit packets towards one base station using the frameless IRSA protocol. Users are grouped in classes of different priorities. The users of one class also share the same degree distribution. As mentioned above, each user has partial observability about the network, because it does not know on which slots the other agents are transmitting (nor

about collisions). However, they have additional information: an important assumption is that the base station is maintaining a discretized estimate $\bar{G}$ of the load $G = \mu$ (Poisson agent arrival rate) in the system and broadcasting it to each agent.

Our objective is to maximize the total throughput of users, for each given network load G; where the throughput of each class is actually weighted by a different factor (so that some classes carry more weight, as a priority mechanism). We assume that each active agent when it decides to send a packet has to send the packet and its replicas within a virtual frame which is associated to the agent. It is interesting to look at the base station perspective: it observes singletons on some slots, collisions on some other slots, and performs SIC for each packet already decoded.

We adapt the Polynomial Weight RM algorithm detailed in Eq. (3) to solve our problem. To emphasize the learning aspect, here, the term "agent" will be used as an equivalent to "user". In order to map the problem features to RM, we have the following assumptions and system model:

• A centralized *offline* learning approach based on Regret Minimization is considered. A large number of simulations or *episodes* (as in Q-Learning) are run. Each episode corresponds to a long contention round. It is intended that after learning has finished, the weights could be used in an actual network, or in our case, are actually used as distributions Λ in further simulations without learning.

• The base station is assumed to broadcast a discretized estimate $\bar{G}$ (with a finite number of possible values) of the actual load G: currently $\bar{G}$ is the measured average number of users per slot since the beginning of the contention round.

• The action of each agent is: selecting the number of repetitions (the degree).

• As per Polynomial Weights RM, each of the agents maintains weight tables (denoted w) which are used to compute the probability of selecting each action, akin to a probability distribution Λ. We extend it: one different table of weights is used depending on some state. The agents consider the load estimate given by the base station as the environment state, and it is discretized to constitute a finite set of possible states. For each different discretized load estimate, the agent uses and updates a different set of weights $(w_i(\bar{G}))_{i \in X}$.

• Additionally, the agents of the same class are sharing the same weight tables w table in the learning process. Updates of the weights after each selected action are thus shared within agents of one class[1]. The goal of using the same w table for all the agents of the same class is to drive the agents inside one class to act cooperatively, and to work in a coordinated manner towards the collective goal. This may depart from usual assumptions in RM and evolutionary dynamics.

• At the moment an agent selects a degree, the results of this action are unknown until some time has elapsed (see Sec. 4.4). Thus an approach with delayed updates similar to *n-step Sarsa* [8, section 7.2] is used.

• The main challenge for applying the Polynomial Weight algorithm is to compute the loss. The loss computation is directly related to the rewards calculation.

[1] Note that then the learning also behaves as if one class would be one agent by itself. The algorithm, and our implementation, also works with non-shared tables.

As our goal is to optimize the joint throughput, we opt to directly link the rewards to the number of decoded agents in each class and set "reward = number of decoded agents". Defining an IRSA reward is otherwise difficult.
• The number of decoded and non-decoded users is available at the global simulator level during our centralized learning process.
We further detail delayed updates and reward computation in the following sections.

4.4 Delayed Updates

Each agent sends within its virtual frame size on the one side, and the base station decodes slot by slot on the other side. Therefore, the base station needs to wait, at least, for the end of the virtual frame to decide if one agent can be decoded or not. Hence, to accurately compute a reward, one delay needs to be introduced: this is illustrated in Fig. 3. As shown in the figure, agent A starts

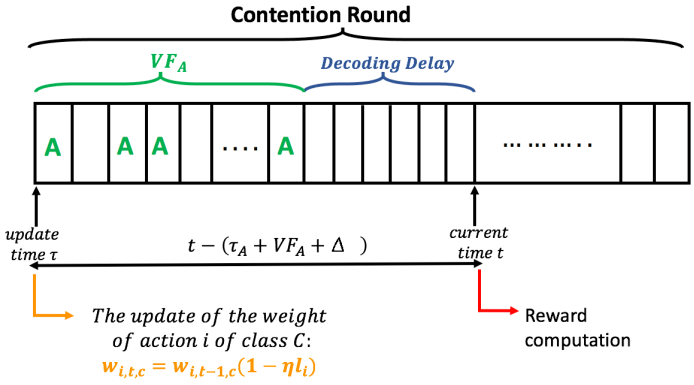


Fig. 3: Reward computation and update delays

to be active at time τ and sends its packets using the action i, e.g. sending i replicas, ($i = 4$ in this case), within the associated virtual frame spanning the time from τ to $\tau + VF_A - 1$. Only at time $\tau + VF_A$, can one be certain that all replicas of A have been sent. But decoding can be further delayed: during this virtual frame, other agents could become active and transmit in overlapping virtual frames (possibly shifted in time, see Fig. 2) and could induce collisions that need to be resolved in order to recover one of the replicas of A. But in turn, those other agents might collide with agents whose virtual frames are occurring even later, etc.[2] For practical purposes, an additional *decoding delay* denoted Δ has to be introduced after which the base station would consider the slots

[2] It is indeed possible to construct a frameless IRSA scenario where one user can be only decoded after an arbitrarily large delay.

definitely non-decodable. As a result, in our simulator, we compute the rewards of an action selected at time τ, only at time $t = \tau + VF_A + \Delta$, and perform the RM weight update at that time. This is similar to n-step Sarsa [8, section 7.2], with $n = VF_A + \Delta$.

4.5 Reward and Loss Computation

We assume that there are two classes C_0 and C_1, and that the class C_0 always has a higher priority than the class C_1. This priority difference is introduced in the reward computation of each class. As the base station decodes the slots up to time t, (see Fig. 3), it computes the number of decoded agents of each class up to the time t. The associated reward of an agent A that played an action i at the time τ is computed at the time t as follows:

$$\begin{aligned} r_i(A) &= P_{C_0,t} && \text{if} && A \in C_0 \\ r_i(A) &= \alpha P_{C_1,t} + (1-\alpha) P_{C_0,t} && \text{if} && A \in C_1 \end{aligned} \tag{4}$$

where:
$r_i(A)$: is the associated reward of action i from the agent A in the class C.
$P_{C,t}$: is the number of decoded packets of agents of class C up to time t.
α: is the parameter that weights the priority of classes.

On Eq. (4), notice that as α is smaller, the priority of class C_0 is higher. Notice also that the reward of each agent is computed based on the collective amount of decoded packets of all agents (of the same class), and hence they act cooperatively. This is in opposition to the selfish behavior of the agents if the reward was based exclusively on the individual performance of each agent.

Now, more importantly, in IRSA, reward computation is difficult, because the decoding process is iterative: it is difficult to assert if an individual action is responsible for undecoded packets. A straightforward reward is used here: essentially the number of decoded packets (or a function of it). In other RL algorithms, this would not work as the reward would grow linearly with time. In RM, however, only the loss (regret) is used for the updates, and it is the difference of reward between the best action and the taken action. In our case, the loss translates as the number of packets that the taken action had prevented from being decoded, which is exactly the meaningful information.

But then in addition to computing the actual reward using Eq. (4), corresponding to the played action i, it is necessary to compute the optimal reward that the agent could receive if it played the optimal action at time τ. This has a cost and increases complexity. In the case of a single-agent system, with no delay in update computation, the optimal reward can also be computed at time t by trying all possible actions at a playing time $t-1$ and considering the action that yields the maximum reward as the optimal action to take at the time $t-1$. If the action space is large, this process could already be costly.

However, it is more complicated in a multi-agent system where 1) reward computation is delayed (here: by necessity), 2) where other agents are also interacting in the environment in the interval between the action of one node and

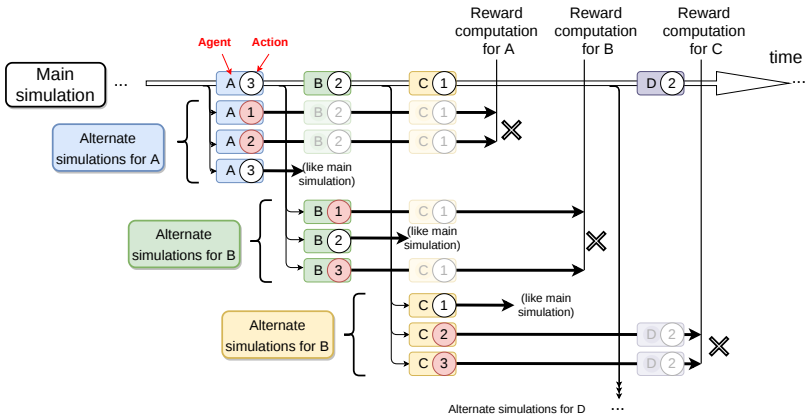


Fig. 4: Alternate simulations for agents A, B, C, and D

its associated reward computation. To handle this, in practice, we maintain one main simulation where each action selected by one agent is actually performed. But we also maintain *alternate simulations* (equivalent to "alternate realities" in mundane terms), that differ from the main simulation only by one action of one agent. Each action of an agent indeed results in creating one new associated alternate simulation for each of its other possible actions (initialized as a copy of the main simulation). At the time of the reward computation for the agent, the reward is computed in each of its alternate simulations: since in its alternate simulations the only difference is the action of the agent (not those of other agents), the difference of reward between different actions can be immediately ascribed to the actions themselves. Fig. 4 illustrates alternate simulations, in a scenario where 3 actions $1, 2, 3$ are possible, and where agent A selects action 3, agent B selects action 2, agent C selects action 1, and agent D selects action 2 in the main simulation. The alternate simulations correspond to simulations where one agent selects each of the 2 alternate actions.

Consider an agent A of class C that selected an action $i \in X$ at time τ.

Its optimal reward is computed as follows:

$$r_{\text{opt}}(A) = \max_{j \in X} r_j(A) \quad \text{with} \quad X = \{0, 1, \ldots, D\} \tag{5}$$

And the loss of playing an action i, by an agent A at time τ is computed using the following equation:

$$\ell_i(A) = \frac{r_{\text{opt}}(A) - r_i(A)}{N} \tag{6}$$

where:

r_i, is the associated reward of action i, which is computed using Eq. (4) and the

knowledge of the class C of node A.
N: is a normalizing factor, taken to be the total number of users in the system.

We summarize the design of our offline regret minimization-based learning algorithm: it is an adaptation of n-step Sarsa [8, section 7.2], where the Q table update is replaced by the weight update from the Polynomial Weights Regret Minimization Eq. (3), and where agents of the same class, share the same weights.

5 Numerical Results

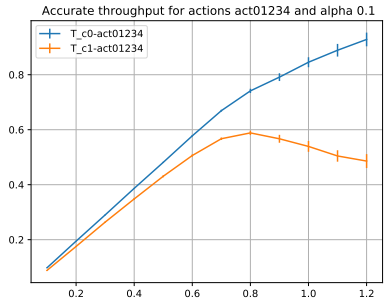
In this section, the performance achieved by IRSA-RM as a random access MAC protocol is illustrated through simulations. There are two phases: first, the learning phase, whose objective is to obtain good degree distributions; second, the performance evaluation of these distributions as is common in IRSA evaluation.

We developed our own simulator for IRSA and RM. For all results, a contention round of $M = 500$ slots is used, the virtual frame size is set to $VF = 150$ slots, while the decoding delay is $\Delta = 50$ slots. For all simulations, both classes have equal arrival rates. The maximum possible degree (action) is $D = 10$. Different cases of class priority are studied; the results are always obtained for two classes, and two different values of the priority parameter: $\alpha = 0.1$ and $\alpha = 0.3$: in both cases, the class C_0 has a higher priority than the class C_1.

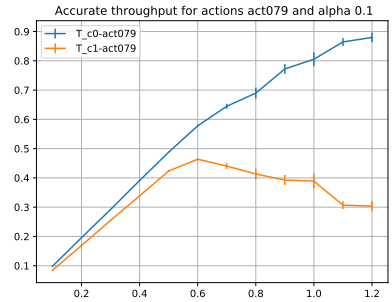
We start with the learning phase, whose objective is to find good degree distributions with respect to Eq. (2), interpreted through Eq. (4). In this phase, agents are restricted to one fixed subset X of actions from the set of all possible actions $X \subset \{0, 1, 2, \ldots, D\}$. Several such subsets are selected. For each action subset, several learning processes are run: the total user Poisson arrival rate $G = \mu$ is fixed during each of them, and one learning process is run for each G taken from 0.1 to 1.2 with step 0.1. At the end of each learning process, for each class C, the RM algorithm yields some weights $(w_{C,i}(\bar{G}))_{i \in X}$ from which probabilities of selecting actions are derived $(p_{C,i}(\bar{G}))_{i \in X}$ which are directly interpreted as lambda distributions (e.g. $\Lambda_i^{\mathrm{RM}}(X, C, \bar{G}) \triangleq p_{C,i}(\bar{G})$). Each learning process is run for $E = 5000$ episodes, and the learning rate η is set to 0.04.

We then compute the performance when applying the obtained distributions. Our main metric is the throughput, e.g. how many decoded packets are recovered per slot. We evaluate the throughput for different loads: but for these simulations, the load does not represent a Poisson arrival rate, but an exact load $g = \frac{N}{M}$, e.g. there are $g \times M$ users exactly, as is common in IRSA performance evaluation. We represent throughput versus load in figures, as is done in [4, fig. 5] for instance.

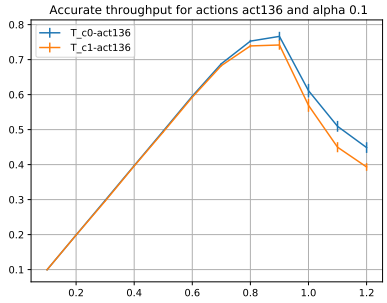
For each selected action set, the throughputs of each of the classes C_0 and C_1 are computed from an average of 300 simulations for a given load g. Note also that for a given load g, one uses the distribution $(\Lambda_i^{\mathrm{RM}}(X, C, \bar{G}))_{i \in X}$ obtained in the learning phase by first selecting the $\bar{G} \in \{0.1, 0.2, 0.3 \ldots 1.2\}$ closer to g. The scaled throughputs are represented in Fig. 5. The scaling factor is 2, to account for the fact that the actual load of one class is $\frac{1}{2}g$, and to make it comparable to classical IRSA (without classes). Thus, the graph for a "perfect" random access protocol would be a line $y = x$ for $x \in [0, 1]$. The priority parameter α is set to



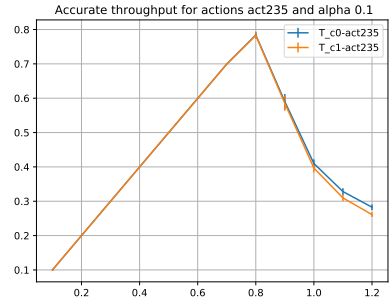
(a) Achieved throughput of both classes, actions [0,1,2,3,4]



(b) Achieved throughput of both classes, actions [0,7,9]



(c) Achieved throughput of both classes, actions [1,3,6]



(d) Achieved throughput of both classes, actions [2,3,5]

Fig. 5: Achieved throughput for frameless IRSA with two classes after using a Regret Minimization based offline learning algorithm

0.1: this means that the agents of class C_1 would trade 10 lost packets of class C_1 for 1 successfully decoded packet of class C_0.

We selected various action sets with different features: some with a continuous set of degrees $(0, 1, 2, 3, 4)$, some with high degrees, some with a mix of both high and low degrees. As the class C_1 tends to a strategy that weights 10 times more than the throughput of the class C_0, we expect it to choose the actions that limit the collisions with the packets of the class C_0, if possible, until around the throughput of class C_0 is somewhere $5\times$ to $10\times$ higher.

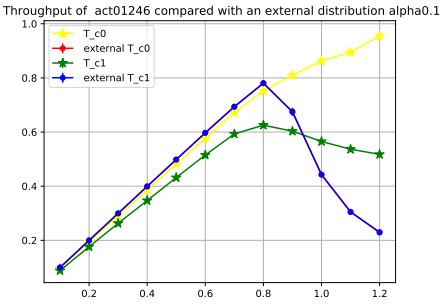
Fig. 5 reports the results for 4 different action sets. We are interested in assessing the quality of the priority mechanism introduced by having different distributions for the classes: it can be measured from the gap between the achievable throughput of the two classes. From the results (confirmed by others not presented here), we find that the first defining feature is the inclusion or not of action 0 in the action set: Fig. 5a and Fig. 5b represent results from two different subsets of actions that include action 0. We can see that the usual sharp decrease

of throughput with IRSA around $g = 1$, does not occur for class C_0: only the throughput of class C_1 decreases at higher loads. The priority mechanism is thus working very well, as class C_1 leaves room for class C_0, indeed as its distribution is: $\Lambda^{\mathrm{RM}}(\{0,1,2,3,4\},1,1.2) = (\Lambda_0 = 0.594, \Lambda_1 = 0.088, \Lambda_2 = 0.086, \Lambda_3 = 0.101, \Lambda_4 = 0.130)$, with $\Lambda_0 = 0.594$, around 60% of its transmissions are suppressed at load $g = 1.2$.

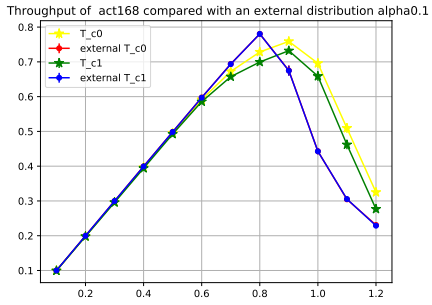
In Fig. 5c and Fig. 5d, we used different actions sets, this time without action 0. We observe that this time, the class C_1 experiences the classical IRSA sharp decrease at a higher load. Introducing action=1 in the set, seems to slightly allow differentiation between classes: for action set $X = \{1, 3, 6\}$, $\Lambda_1(C_0) = 0.412$ and $\Lambda_1(C_1) = 0.590$, therefore a noticeable amount of packet transmission is just one single transmission (e.g. no repetition). Transmissions with such degree=1 colliding on the same slot cannot be retrieved by SIC, hence automatically result in lost packets (and lost slots). Therefore, at higher loads, the protocol has to find a balance between this phenomenon (wasting slots), and higher degree repetitions, that can benefit from SIC, but also risk blocking a number of slots, if undecoded. Action 1 appears safer, from shown values of Λ_1.

In [4], a framework for finding degree distributions was proposed for framed IRSA, using Density Evolution (for deterministic performance evaluation), and using Differential Evolution (as a heuristic for finding a solution of Eq.(1)): this method aims to find the distribution with the highest load threshold G^*, that is, the load up to which packet loss is vanishingly small when frame size increases towards infinity. These distributions are good comparison points, even though they are optimized for a different context. Fig. 6 shows the comparison between the achieved throughput by IRSA-RM with two classes and the achieved throughput by using the IRSA degree distribution $\Lambda_2 = 0.5, \Lambda_3 = 0.28, \Lambda_8 = 0.22$ from [4] (named there "$\Lambda_3(x)$") which we refer to as "external distribution". Fig. 6a shows a higher achieved throughput for the class C_0 using the learned set of actions $\{0, 1, 2, 4, 6\}$ with IRSA-RM compared to an external distribution. This is due to the priority mechanism: using the action 0 a sizeable amount of time from the class C_1 leaves free slots for the class C_0. This same effect appears in Fig. 6b, thanks to degree $= 1$. In contrast, the achieved throughput for both classes in Fig. 6c and Fig. 6d is comparable to throughput of the external distribution (always close, and for load $g > 0.8$, better for $\frac{1}{4}$ of the points, otherwise worse). Both IRSA-RM and and the external distribution achieve the same maximum load 0.8. This comparison proves that our learning algorithm operates very well in its objective of finding good distributions.

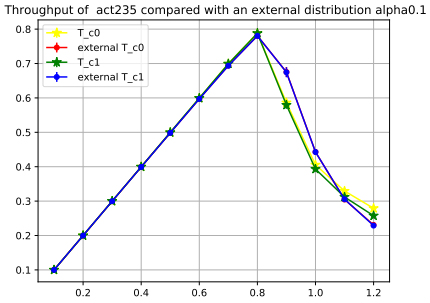
Next, we study the impact of the priority parameter α on the achieved throughput in both classes. In Fig. 7a, we compare the achieved throughput of both classes where $\alpha = 0.1$ and $\alpha = 0.3$ when using the action set $\{0, 7, 9\}$. As previously in Fig. 5a, with $\alpha = 0.1$, and action 0 in the action set, the priority mechanisms work well. The gap between the achievable throughput of both classes decreases (in blue) when α is increased to 0.3, as expected. Indeed $\Lambda_0(C_1, g = 1.2)$ decreases from 0.759 to 0.591 (hence action 0 is less used). When action 0 is not available, as in Fig. 7b with action set $\{1, 3, 5\}$, the best option for the class C_1 to increase the throughput of the other class is to choose the action



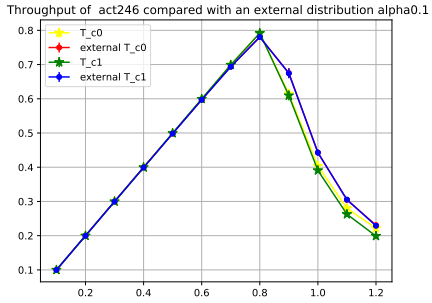
(a) Throughput comp. between actions $\{0, 1, 2, 4, 6\}$ and an external distribution



(b) Throughput comparison between actions $\{1, 6, 8\}$ and an external distribution



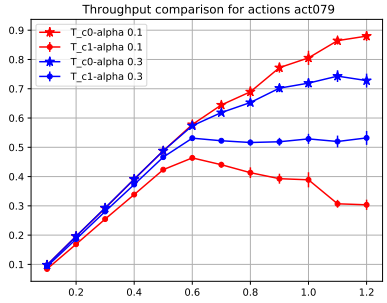
(c) Throughput comparison between actions $\{2, 3, 5\}$ and an external distribution



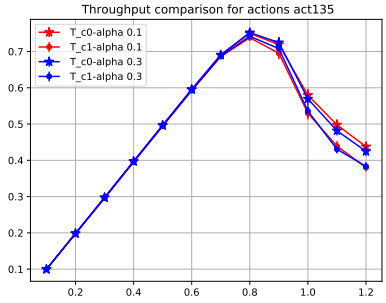
(d) Throughput comparison between actions $\{2, 4, 6\}$ and an external distribution

Fig. 6: Throughput comparison between different set of actions and an external distribution ($\Lambda_2 = 0.5, \Lambda_3 = 0.28, \Lambda_8 = 0.22$)

1 (as $\Lambda_1(C_1) = 0.594$ for $g = 1.2, \alpha = 0.1$). As explained previously, the impact is still limited as shown by the small gap, and small difference when $\alpha = 0.3$ (and $\Lambda_1(C_1) = 0.538$ for $g = 1.2, \alpha = 0.3$). Indeed, the class C_1 has no other choice than to send at least one replica, which will always occupy some slot(s). Finally, Fig. 8 reports the convergence of the RM learning process. The learning parameter was set to $\eta = 0.04$. Recall that the learning algorithm updates the weights of the actions $(w_i)_{i \in X}$ for each selected action after the proper update delay, and that these weights are used to compute the probabilities $(p_i)_{i \in X}$ of selecting each action according to Eq. (3). Again, these are equivalent to a degree distribution Λ. In Fig. 8a, for a network load $G = 0.8$ and $\alpha = 0.1$, we show the evolution of the probabilities during learning for the action set $\{0, 1, 3, 6\}$ at the end of each episode. The probabilities of selecting the smaller degrees 0 and 1 are dropping while the probabilities to use the larger degrees 3 and 6 are rising. The changes stop around episode 3300 where the probabilities start to plateau (it is also true for class C_1). Disregarding action 0 (and to some extent,

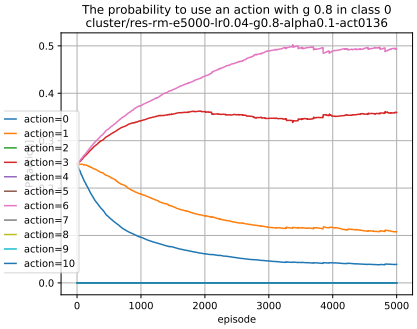


(a) Throughput of actions $\{0, 7, 9\}$ for $\alpha = 0.1$ and $\alpha = 0.3$

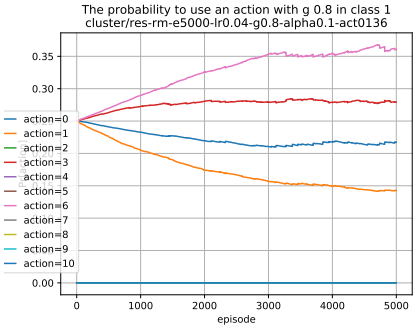


(b) Throughput of actions $\{1, 3, 5\}$ for $\alpha = 0.1$ and $\alpha = 0.3$

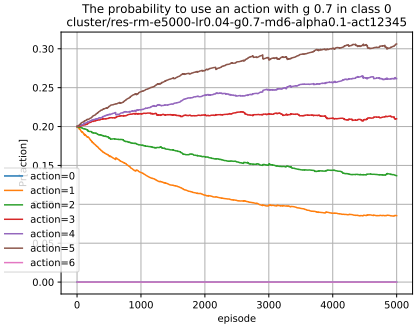
Fig. 7: Throughput comparison for different set of actions and different priority parameter values



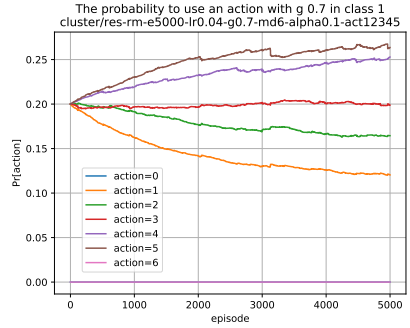
(a) The convergence of the probabilities of taking the actions $\{0, 1, 3, 6\}$ for class C_0



(b) The convergence of the probabilities of taking the actions $\{0, 1, 3, 6\}$ for class C_1



(c) The convergence of the probabilities of taking the actions $\{1, 2, 3, 4, 5\}$ for class C_0



(d) The convergence of the probabilities of taking the actions $\{1, 2, 3, 4, 5\}$ for class C_1

Fig. 8: The convergence of the probabilities of taking the actions for both classes and for different sets of actions

action 1) is the result of class C_0 attempting to maximize its throughput. On the other hand, probabilities of action 0 and 1 have the inverse behavior for class C_1; notice that because G is not so high, action 0 is still not the most selected. In Fig. 8c and Fig. 8d, we show the convergence of the probabilities for another set of actions without action 0 and for a network load $G = 0.7$ and $\alpha = 0.1$. The probabilities show a form of convergence around episodes $3100 - 3200$ for both classes. Notice that the learning rate could be a function of the episodes as in 1/(episode index), but for practical purposes, our fixed learning parameter appears sufficient for our learning phase.

6 Conclusion

In this article, we studied one of the modern random access protocols: Irregular Repetition Slotted Aloha (IRSA) in its frameless version. We adapted a reinforcement learning approach based on Regret Minimization to optimize the transmission strategy of this protocol, and thus proposed the protocol "IRSA-RM". RM is well suited to IRSA, as in both cases, one uses a set of probabilities of selecting a given number of repetitions Λ. The learning is performed offline: it learns the main protocol parameters (the user degree distribution) for a set of predefined network loads. After the learning phase, the parameters can be later used in a network: assuming that the estimate of the load is broadcast by the base station, each device will select the set of parameters that were learned with the closest load. We detailed precisely the mapping between our problem, optimizing IRSA, and the centralized learning approach with RM, including delayed updates, reward computation, and alternate simulations, the introduction of priority classes, etc. Simulation results show a very high level of performance of IRSA when it is optimized with Regret Minimization, and how IRSA-RM behaves for different types of actions (degrees) sets. Future work will include considering richer actions, more sophisticated RM techniques such as CFR, and applying Deep Reinforcement Learning techniques.

References

1. Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. Evolutionary Dynamics of Multi-Agent Learning: A Survey. *JAIR*, 53:659–697, August 2015.
2. Avrim Blum and Yishay Mansour. Learning, Regret Minimization, and Equilibria. In Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*, pages 79–102. Cambridge University Press, 2007.
3. Y. Chu, P. D. Mitchell, and D. Grace. ALOHA and Q-Learning based medium access control for Wireless Sensor Networks. In *Proceedings of ISWCS 2012*, pages 511–515, August 2012. ISSN: 2154-0225.
4. Gianluigi Liva. Graph-Based Analysis and Optimization of Contention Resolution Diversity Slotted ALOHA. *IEEE Trans. Commun.*, 59(2):477–487, February 2011.
5. Enrico Paolini, Gianluigi Liva, and Marco Chiani. Coded Slotted ALOHA: A Graph-Based Method for Uncoordinated Multiple Access. *IEEE Trans. Inform. Theory*, 61(12):6815–6832, December 2015.

6. C. R. Srivatsa and C. R. Murthy. Throughput Analysis of PDMA/IRSA under Practical Channel Estimation. In *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5, July 2019. ISSN: 1948-3252.
7. C. Stefanovic, P. Popovski, and D. Vukobratovic. Frameless ALOHA Protocol for Wireless Networks. *IEEE Commun. Lett.*, 16(12):2087–2090, December 2012.
8. Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, second edition, 2018.
9. S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari. Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks. *IEEE Trans. Cogn. Commun. Netw.*, 4(2):257–265, June 2018.
10. Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret Minimization in Games with Incomplete Information. page 8.
11. Casini,E., De Gaudenzi, R. and Del Rio Herrero, O. "Contention Resolution Diversity Slotted ALOHA (CRDSA): An Enhanced Random Access Scheme for Satellite Access Packet Networks", IEEE Trans. Wirel. Commun., vol. 6, no. 4, pp. 1408-1419, April 2007.
12. Fooladivanda,D., Al Daoud,A., Rosenberg,C. "Joint Resource Allocation and User Association for Heterogeneous Wireless Cellular Networks", IEEE Transactions on Wireless Communications, 2011, vol. 12, pp. 384-390.
13. Ge,X., Li, X., Jin,H., Cheng,J. and Leung,V.C.M. "Joint User Association and User Scheduling for Load Balancing in Heterogeneous Networks",IEEE Transactions on Wireless Communications, 2018, vol. 17, pp. 3211-3225.
14. Luong, N. C. et al. "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey", IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3133-3174, Fourthquarter 2019,
15. Naparstek,O., Cohen,K. "Deep Multi-User Reinforcement Learning for Distributed Dynamic Spectrum Access", IEEE Trans. Wirel. Commun., 2019, vol. 18, pp. 310-323.
16. Destounis,A., Tsilimantos,D., Debbah, M. and Paschos,G.S. "Learn2MAC: Online Learning Multiple Access for URLLC Applications," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 2019, pp. 1-6.
17. Toni, L. and Frossard, P. "IRSA Transmission Optimization via Online Learning", 2018.
18. Wang, L., Xiao J. and Guanrong, C. "Density evolution method and threshold decision for irregular LDPC codes", International Conference on Communications, Circuits and Systems (IEEE Cat. No.04EX914), Chengdu, 2004, Vol.1, doi: 10.1109/ICCCAS.2004.1345932, pp. 25-28.
19. Hmedoush, I., Adjih, C., Mühlethaler, P. and Kumar, V., "On the Performance of Irregular Repetition Slotted Aloha with Multiple Packet Reception", 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 2020, pp. 557-564, doi:10.1109/IWCMC48107.2020.9148173.
20. Nguyen,T. T., Nguyen N. D. and Nahavandi,S., "Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications", IEEE Transactions on Cybernetics, vol. 50, no. 9, pp. 3826-3839, Sept. 2020
21. Klos, T., Jan van Ahee , G., and Tuyls, K., "Evolutionary Dynamics of Regret Minimization", Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part II. 2010,Vol.6322, pp. 82-96